



# Praktische Informatik 3: Funktionale Programmierung

## Vorlesung 4 (08.11.2022): Übungen

Christoph Lüth



Deutsches  
Forschungszentrum  
für Künstliche  
Intelligenz GmbH



Universität  
Bremen

Wintersemester 2022/23

# Jetzt seid ihr dran!

## Übung 4.1: Neue Typen

Sind folgende Ausdrücke typkorrekt, und wenn ja welchen Typ haben sie?

- ① `right (Pair (3 + 4) Empty)`
- ② `head (Pair (Cons 'x' Empty) True)`
- ③ `right (head (Cons (Pair 'x' 3) Empty))`
- ④ `head (tail (Cons 3 (Cons 4 Empty)))`

# Jetzt seid ihr dran!

## Übung 4.1: Neue Typen

Sind folgende Ausdrücke typkorrekt, und wenn ja welchen Typ haben sie?

- 1 `right (Pair (3 + 4) Empty)`
- 2 `head (Pair (Cons 'x' Empty) True)`
- 3 `right (head (Cons (Pair 'x' 3) Empty))`
- 4 `head (tail (Cons 3 (Cons 4 Empty)))`

Lösung:

- 1 Typ: `List  $\alpha$`
- 2 Typfehler
- 3 Typ: `Integer`
- 4 Typ: `Integer`

# Jetzt seid ihr dran!

## Übung 4.2: Vordefinierte Typen

Sind folgende Ausdrücke typkorrekt, wenn ja welchen Typ haben sie, und was ist ihr Wert?

- ① `take 4 (replicate 3 (3, 4))`
- ② `snd (unzip (zip [1..10] "foo"))`
- ③ `"a" ++ [('a')]`
- ④ `head [("foo", []), ("baz", 4 :: Integer)]`

# Jetzt seid ihr dran!

## Übung 4.2: Vordefinierte Typen

Sind folgende Ausdrücke typkorrekt, wenn ja welchen Typ haben sie, und was ist ihr Wert?

- ① `take 4 (replicate 3 (3, 4))`
- ② `snd (unzip (zip [1..10] "foo"))`
- ③ `"a" ++ [('a')]`
- ④ `head [("foo", []), ("baz", 4 :: Integer)]`

Lösung:

- ① Typ: `[(Integer, Integer)]`, Wert: `[(3,4), (3,4), (3,4)]`
- ② Typ: `String`, Wert: `"foo"`
- ③ Typ: `String`, Wert: `"aa"`
- ④ Typfehler

# Jetzt wieder ihr!

## Übung 4.3: Meine Paare

Erinnert auch an die selbstgemachten Paare?

```
data Pair  $\alpha$   $\beta$  = Pair { left ::  $\alpha$ , right ::  $\beta$  }
```

Schreibt eine `Show`-Instanz, welches ein Tupel als `(a, b)` anzeigt!

# Jetzt wieder ihr!

## Übung 4.3: Meine Paare

Erinnert auch an die selbstgemachten Paare?

```
data Pair  $\alpha$   $\beta$  = Pair { left ::  $\alpha$ , right ::  $\beta$  }
```

Schreibt eine `Show`-Instanz, welches ein Tupel als `(a, b)` anzeigt!

Lösung:

- ▶ Voraussetzung: `Show a`, `Show b`
- ▶ Klammersetzung beachten

```
instance (Show a, Show b)  $\Rightarrow$  Show (Pair a b) where  
    show (Pair a b) = "(" ++ show a ++ ", " ++ show b ++ ")"
```